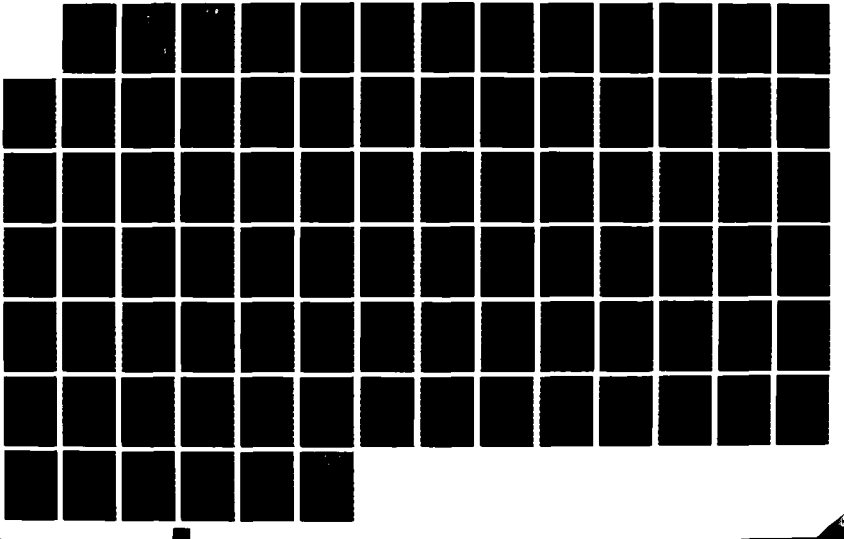
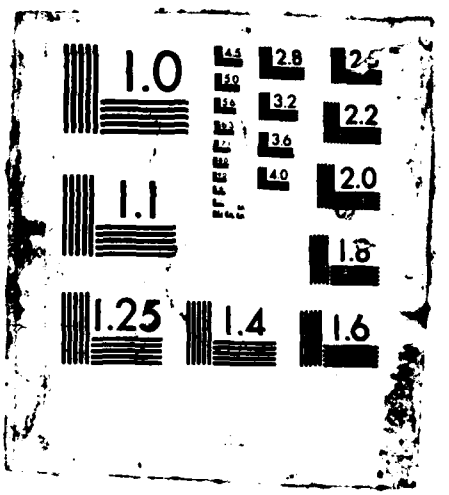
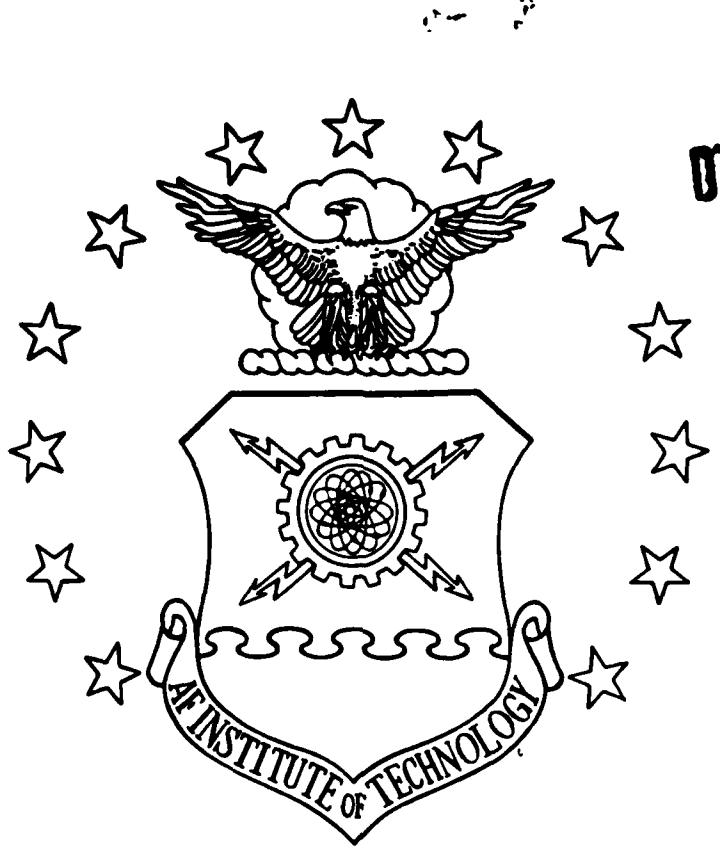
AD-A170 873 A PROTOTYPE KNOWLEDGE-BASED SYSTEM FOR SATELLITE MISSION PLANNING(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGINEERING 1/1

UNCLASSIFIED D E PERALES DEC 86 AFIT/GE/ENG/86D-17 F/G 22/1 NL




1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.8
3.2
3.6
4.0



A PROTOTYPE KNOWLEDGE-BASED SYSTEM FOR SATELLITE MISSION PLANNING

THESIS

David E. Perales
Captain, USAF

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GE/ENG/86D-17

A PROTOTYPE KNOWLEDGE-BASED SYSTEM FOR SATELLITE MISSION PLANNING

THESIS

David E. Perales
Captain, USAF

AFIT/GE/ENG/86D-17

Approved for public release; distribution unlimited

AFIT/GE/ENG/86D-17

A PROTOTYPE KNOWLEDGE-BASED SYSTEM FOR SATELLITE MISSION PLANNING

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
Air University
In Partial Fulfillment of the
Requirements for the Degree of
Master of Science in Electrical Engineering

David E. Perales, B.E.E.
Captain, USAF

December 1986

Acknowledgements

There are several people who deserved recognition for their support and help during the duration of this thesis project. I am deeply indebted to thank LtCol Cochran and the Air Force Satellite Control Facility for showing interest in this project and sponsoring it. Special thanks is extended to Maj Woffinden for his insight into what was expected out of a thesis. A debt of gratitude is owed to LtCol Parnell, my thesis advisor, for his guidance during the formulation of this project. His influence has enabled me to reach my full potential. However, my greatest appreciation goes to Shirley, my wife, for all her patience, support, and advice during the past 18 months. Despite both of us working on our master degrees and sharing our home computer for our theses, our marriage has survived the assignment at AFIT and grown stronger in the process.

David E. Perales

TABLE OF CONTENTS

List of Figures

List of Tables

Abstract

The growing complexity of modern artificial satellites and limited resources available for satellite operations has caused satellite mission planning to become a data intensive job which overwhelms mission planners. The purpose of this ~~research~~ Thesis was to determine the feasibility of using Artificial Intelligence techniques, specifically knowledge-based systems, in satellite mission planning.

Research was conducted to determine the type of knowledge representation that best accommodates data intensive problem domains. Using this basis, a prototype knowledge-based system for use on a microcomputer was designed, constructed, and evaluated. The satellite schedule was generated based on prioritized user requirements, the requirements' acquisitions, and available resources. Explanations were provided to enable the mission planner to understand how the schedule was generated and allow him to make changes as user requirements change.

The prototype system showed that a knowledge-based system can assist the mission planner in scheduling satellite operations. This establishes a base from which more research can be done to help determine the optimal environment required to support an operational mission planning knowledge-based system.

A PROTOTYPE KNOWLEDGE-BASED SYSTEM FOR SATELLITE MISSION PLANNING

I. Introduction

Background

During the 1960's, the use of satellites was in its infancy. The mission planning for those satellites consisted of turning the satellite on and off at desired times of operation. Today, modern satellites have increased in complexity and satellite users are insisting that more requirements be satisfied while using fewer resources. Therefore, much more careful and accurate planning is required.

Satellite mission planning is composed of two activities: analysis of user requirements and satellite scheduling. The analysis of user requirements includes understanding and interpreting requirements which may not be explicitly stated. In addition, the analysis requires understanding the health and operation of the satellite. This analysis allows the schedule to be tailored and insures as many requirements as possible are scheduled. These activities are time consuming events that depend on the knowledge and experience of expert mission planners. Satellite scheduling consists of arranging satellite operations so that user requirements are satisfied and the health of the satellite is maintained. Application of artificial intelligence techniques to satellite mission planning can cut time, cost, and risk (26:50).

In summary, the mission planner has three basic areas of concern: understanding and interpreting user requirements, understanding the operation and maintaining the health of the satellite, and scheduling the satellite to meet as many requirements as possible with limited constraints. By using knowledge-based system technology, a mission planning system can improve efficiency because the satellite schedule is generated quicker using less manpower. The Jet Propulsion Lab conducted tests with the system Deviser III and showed that this improvement in time can be 10-50 times faster than human mission planners (26:50). In addition, the effectiveness of mission planning is improved because lower priority requirements have an increased probability of getting scheduled when the knowledge-based system is used.

Problem

For satellite scheduling, the complexity of satellites and their mission requirements are such that a human mission planner can only do an average job of optimizing the limited resources (tape recorders used to record data when the satellite is not in view of a tracking station, satellite power, and ground tracking facility equipment on earth) while satisfying only some of the user requirements. As satellites increase in complexity, consideration needs to be given to artificial intelligence-based expert systems that have been designed to execute tasks without preprogramming all conceivable paths (8:77). This thesis project takes an initial look at the

feasibility of using a knowledge-based system to assist the mission planner in satellite operations.

Scope

As mentioned earlier, the mission planner has three areas of concern. This thesis takes an initial look at the feasibility of using a knowledge-based system to assist the mission planner in the area of satellite scheduling. Satellite scheduling is composed of procedures and rules used for scheduling the greatest number of requirements possible within a given time span. A prototype knowledge-based system that automatically schedules prioritized user requirements was designed and implemented in this thesis. In addition to scheduling requirements, the system provides explanations of the system's actions with respect to each requirement.

Assumptions

This thesis is based on three assumptions. The first is that satellite systems are composed of a satellite which has limited tape, power, and payload configurations, and tracking stations which are required to communicate with the satellite. The second is that actual requirements for a satellite program can be prioritized. The third is that the parameters (name, duration, payload, acquisition time, etc) used in the requirements in this thesis approximate those used in an operational system. These assumptions allowed the designer to concentrate on making sure that the prototype correctly:

1. Schedules by priority if resources are available.

2. Gives proper reasons for requirements not being scheduled.

General Approach

The following seven steps were taken in developing the prototype knowledge-based system:

1. Research was conducted to determine how other mission planning and scheduling systems represented their knowledge-bases and what control strategies were used. Information gathered from the research was reviewed to determine if there were any commonalities that correspond to systems of this type.
2. Procedures and methods used by mission planners to plan and schedule satellite operations were specified.
3. The knowledge representation and the control strategy were selected.
4. System requirements were described for an operational system. The requirements were then reevaluated within the constraints of a thesis effort to lead to a prototype system.
5. The prototype was designed and coded based on the modified requirements from step 3.
6. The prototype was tested and evaluated to determine if it met the system requirements defined in step 3.
7. Modifications were made to the prototype to correct any discrepancies that were identified. In addition, minor design modifications were made to make the prototype meet system requirements.

8. Steps 6 and 7 were repeated to construct the prototype knowledge-based system. This evolutionary development technique has emerged as the dominant expert/knowledge-based system methodology (13:23).

Materials and Equipment

The prototype knowledge-based system designed in this thesis was implemented in PC Scheme, a microcomputer implementation of Texas Instrument's Scheme. Selection of PC Scheme is discussed in the system requirements section in Chapter IV. PC Scheme includes the Scheme Object-Oriented Programming System (SCOOPS) which allows the implementation of a frame-based knowledge-based system. The schedule timeline graphics were implemented by interfacing TURBO Pascal with PC Scheme.

The software was developed and evaluated on two hardware systems. The first system was an AT&T 6300 with a 20 MB hard disk and the second was an IBM PC AT computer. Both systems included a graphics capability.

The materials and equipment required for this thesis were available from the onset of the thesis. No other support was required to conduct this thesis.

Overview

This first chapter provided a brief presentation of the background and assumptions pertaining to this thesis. In addition, a description of the problem, scope, and approach used in this thesis was given.

The second chapter reviews current scheduling systems. The features of each system are presented and commonalities are highlighted.

The third chapter discusses satellite mission planning. The objectives, inputs, and constraints of mission planners are defined and examined. Examples are used when applicable to show how the thesis relates to actual operations.

The fourth chapter presents the system requirements. First, system requirements are defined for the system's hardware and software. Second, the system requirements for the knowledge-base are presented. Finally, the human-computer interface for the prototype is discussed.

The fifth chapter describes the design of the prototype system, including module descriptions.

This sixth chapter analyzes and evaluates the prototype system designed in this thesis.

The seventh and final chapter summarizes the thesis effort, presents conclusions, and offers recommendations for future research including system enhancements and suggestions for follow-on thesis work.

Appendix A is a short user's manual that gives future users an insight into how to operate the prototype and some of the limitations of the system.

Appendix B presents how a fully operational knowledge-based system may be used by different groups in an operational environment.

II. Literature Review

As management science has recognized, it is not practical to separate planning from scheduling (9:15). Satellite mission planning is a good example since one specific phase of the planning is satellite scheduling. Since this thesis is concerned with scheduling, several scheduling systems were reviewed to analyze their features and to note any commonalities between the systems. These systems are summarized in Table I. The table is divided into selected categories which describe the systems. System details are written into each category when the information for the specific system was available. The first column states the system name and the type of scheduling the system performs. The stage of development and the agencies where the systems are developed is presented in the second column. The third column presents the type of software and hardware environments that support the systems. The type of knowledge representation used in the database is indicated in the fourth column. Finally, the last column presents some key areas the systems support and/or some capabilities which are built into the systems.

ISIS

In the thesis "Constraint-Directed Search: A Case Study of Job-Shop Scheduling" by Mark Fox, the scheduling system ISIS is presented (9). ISIS is a scheduling system that was designed for the job-shop scheduling domain. The goal of ISIS is

Table I. Selected Scheduling Systems

NAME/ SCHEDULING TYPE	STAGE OF DEVELOPMENT/ RESPONSIBLE AGENCY	SOFTWARE/ HARDWARE ENVIRONMENT	KNOWLEDGE REPRESENTATION	DESCRIPTION
ISIS Job-Shop	Prototype Carnegie-Mellon University	Schema Representation Language DEC VAX 11/780	Frames / Rules	- addresses constraint satisfaction and relaxation - handles changes in plant status - suggests alternative schedules
GENSCHED Job-Shop	In use / Prototype Georgia Tech Research Institute	Zetalisp flavor system	Frames / Rules	- manual and automatic scheduling - "what-if" processing of orders
KAOS Flight Planner	In use NASA Ames Research Center	MRS (Meta-Level-Reasoning-System) VAX 11/780 (operational) Symbolic 3600 (development)	Rules	- addresses constraint satisfaction
SPOT Resource Utilization	Prototype SRI International	SIPE (System for Interactive Planning and Execution monitoring)	Rules	- user can watch, guide, or control planning process - addresses the cooperative process between computer and decision maker
NUDGE Office	Prototype Stanford Research Institute	INTERLISP under TENEX PDP-10	Frames / Rules	- addresses knowledge-base vs power-base AI programs

to construct schedules which satisfy as many constraints as possible in near realtime.

There are several key components of ISIS. First, the knowledge-base representation is a frame-based system that models constraints as well as the organization.

Second, ISIS introduces a number of new concepts in the area of search:

1. A general representation for constraints including relaxations, interactions, and obligations.
2. Constraint-directed bounding of the solution space.
3. The generation and evaluation of constraint relaxation during the search process.
4. Techniques for diagnosing poor schedules.

Finally, ISIS represents a system that for the first time can represent and consider all the domain constraints during the construction of a schedule. In addition, ISIS has been created with all the facilities required for practical use in the factory.

GENSCHED

In "GENSCHED - A Real World Hierarchical Planning Knowledge-Based System" by Semeco, GENSCHED is described as a hierarchical planning system designed to schedule production orders in manufacturing facilities (22). The system is composed of three components; the data entry subsystem, the data display subsystem, and the hierarchical planning subsystem. The data entry subsystem allows manual and semi-manual scheduling and "what-if" processing of orders. The data display

subsystem provides the user immediate feedback to the effect of the changes made to the schedule. In addition, the data display subsystem presents the schedule graphically with precedence and response conflicts being highlighted. The final subsystem, the hierarchical planner, uses the repetitive nature of the plans to efficiently generate valid schedules.

KAOS

The system KAOS (Kuiper Airborne Observatory Scheduler) is presented in NASA's technical memorandum "A Knowledge-Based Expert System for Scheduling of Airborne Astronomical Observations" (18). KAOS is a knowledge-based system designed to assist in route planning of a C-141 flying astronomical observatory.

The user inputs the astronomical bodies and viewing durations he wishes to observe. KAOS then generates flight plans by conducting a search of bodies that satisfy the constraints. The scheduling is accomplished by generating one leg of the flight plan and then testing it. Observations are usually rejected if they violate one of the following constraints:

1. The object is outside the window.
2. The leg overflies a restricted zone.
3. The leg overflies a warning zone.
4. The leg leads to a point that the aircraft runs out of fuel.

Any of the constraints can be relaxed except for the first constraint. Relaxing the first constraint would violate the input requirement of the user.

SPOT

Robinson and Wilkins discusses SPOT in the article "Man-Machine Cooperation for Action Planning" (21). This system is designed to assist in planning the movement and launching of planes on a carrier. SPOT generates plans automatically but has the capability to allow the user to interactively guide and/or control the planning process. Planning for the system is performed hierarchically from general to specific actions with the actions being performed sequentially or in parallel. The user interface is given a great deal of consideration with a display graphically containing the planned carrier deck configuration.

NUDGE/BARGAIN

In the article "Using Frames in Scheduling", Goldstein and Roberts summarizes NUDGE as a knowledge-based office scheduling program in which the actual scheduling is handled by the program called BARGAIN (10). NUDGE accepts informal scheduling requests and produces a schedule containing conflicts and a set of strategies for conflict resolution. BARGAIN, a domain independent search algorithm program, uses traditional decision analysis techniques to control the search process involved in conflict resolution and scheduling.

The knowledge representation in NUDGE consists of frames. When informal requests are received by the system, a frame structure is generated from a set of generic frames. Information missing in the request is computed from defaults,

constraints, and procedures associated with these generic frames.

BARGAIN accepts the schedule containing conflicts from NUDGE and resolves the conflicts individually using resource-driven or purpose-driven conflict resolution techniques. The resource-driven strategy attempts to reschedule the particular time interval while maintaining the event requirement. The purpose-driven strategy attempts to analyze the goal of the event requirement and modify or delete requirements of lower priority.

Summary

The scheduling systems presented point out several key concepts and trends that should be kept in mind when designing a scheduling system.

1. Most of the systems' knowledge representations are framed-based. As their structure suggests, frame systems are useful for problem domains where expectations about the form and content of the data play an important role in problem solving (27:74-75). Scheduling satellite operations falls into this problem domain.
2. Most of the systems demonstrate that constraints play a major role in generating a schedule. Some systems have provides facilities for context-dependent constraint relaxation.
3. All systems indicate that conflicts will occur and methods have to be created to handle the conflicts.

Resource-driven or purpose-driven conflict resolution (NUDGE/BARGAIN) are two ways that conflicts may be resolved.

4. All systems demonstrate that when generating a schedule, a method that bounds the generation of a desired scheduled (i.e. constraint-directed search) should be specified.
5. Most of the systems are prototypes that have not been used to generate operational schedules.
6. All of the systems were supported by hardware environments that consisted of at least a minicomputer.

With these concepts in mind, an analysis of the task domain for satellite mission planning can be accomplished.

III. Analysis of the Task Domain

Everyone wants their system to be intelligent. Waterman proposes this method:

> To make a program intelligent, provide it with lots of high-quality, specific knowledge about some problem area. This realization led to the development of special-purpose computer programs, systems that were expert in some narrow problem area. (28:4)

This thesis implemented a prototype knowledge-based system in the problem domain of satellite mission planning. This chapter summarizes satellite mission planning and provides a basis for discussion of system requirements. Where applicable, examples are given using a weather satellite. The mission of this satellite is to take pictures and sensor readings of weather all over the world.

Satellite Operations

Once a satellite is in orbit, the mission planner must insure the accomplishment of the satellite's mission while maintaining the health of the satellite.

The satellite's mission is to satisfy as many of the user requirements as possible. In order for the mission planner to do this, he needs to understand several areas of satellite operations. First, he must understand the user requirements and know how to use requirement alternatives or modifications that allow the requirement to still be satisfied. Second, the mission planner must take into account the satellite's orbit, the satellite's capabilities and resources, and tracking station resources. By understanding these areas, the mission

planner can maximize the use of the satellite and optimize use of the available limited resources.

Mission Planning Constraints

As previously mentioned, there are several constraints that the mission planner must consider when scheduling user requirements. These constraints are the specifications of the user requirements, the satellite orbit, the satellite's capabilities, and the available tracking stations. Each of these constraints is discussed in detail in the following sections.

User Requirements

User requirements are requests from the community of users which specify a desired operation of the satellite. Each requirement contains all the information essential to satisfy the user's needs. The following list presents the type of information included and an explanation for each:

1. Requirement Name: The name of the requirement, which should give some indication of the type of requirement.
2. Area of Interest: The location, area, or coordinates of the area of interest.
3. Requirement Start Date: The date the mission planner starts to attempt to schedule the requirement.
4. Requirement Duration: The duration the requirement is active. The duration may be given in hours, days, weeks, or months.

5. Requirement Priority: The priority of the requirement with respect to all the other requirements for the specified satellite. This priority is assigned by the satellite's program office. 1 is considered the highest priority with 100 being the lowest. Several user requirements may have the same priority, in which case the scheduling order is determined by which requirement the scheduler encounters first in the data base.
6. Minimum Acquisition Duration: The minimum time the user requires any satellite operation to occur for their specific requirement.
7. Payload Configuration: The desired configuration that the satellite must be in when satisfying the user's request.

The mission planner understands this information and knows what can be changed. An example using a weather satellite helps to clarify the information contained in a user requirement.

The Navy requires the satellite to observe the weather in the middle of the Atlantic Ocean during an exercise. The requirement might look like:

1. Requirement Name: Naval exercise "Tough Exercise"
2. Area of Interest: Latitude - 25N Longitude - 40W
   Circle of 100 mile radius
3. Requirement Start Date: 14 August 1987
4. Requirement Duration: 1 month
5. Requirement Priority: 3

6. Minimum Acquisition Duration: any duration acceptable

7. Payload Configuration: visual sensor

A mission planner would interpret this information to indicate that the Navy wants the visual sensor covering an area centered at latitude - 25N ,longitude - 40W and having a radius of 100 miles. The mission planner plans to satisfy this requirement starting on 14 Aug 1987 and attempts to schedule it for one month. The requirement is scheduled when it does not conflict with any number 1 and 2 priority requirements, there are satellite resources available, and an acquisition of the area of interest occurs.

Satellite Orbit

When a mission planner attempts to schedule requirements, he is limited to satellite acquisitions of the areas of interest. The acquisitions for the areas of interest are determined by the orbit of the satellite. For a geosynchronous satellite, the visible areas on earth are limited, but they can be seen 100% of the time. For an orbiting satellite, all areas on earth are accessible, but only periodically.

Satellite Capabilities and Resources

The mission planner is also constrained by the satellite. He is constrained by the possible payload configurations of the satellite. In addition, the satellite has limited resources (power and tape recorders) which the mission planner needs to optimize in order to schedule the maximum number of requirements. The following sections describe and give examples of

satellite limitations for payload configurations, tape recorders, and power.

Payload Configurations. Satellites are designed and constructed to meet certain mission requirements. Once in orbit, the mission planner is constrained to only predetermined configurations. Some configurations are compatible while other configurations must be alternated between each other. An example shows why the mission planner needs to understand payload configurations.

Suppose that an acquisition for the requirement previously stated (the naval requirement "Tough Exercise") occurs at the same time as a lower priority requirement. The lower priority requirement requires the satellite to look at Puerto Rico with the infrared sensor. The pointing angles for the two requirements are 120 degrees apart, and the satellite has a view of only 45 degrees. The mission planner would normally select the naval exercise requirement because it has a higher priority and the satellite cannot look at both requirements at the same time. However, by understanding the payload configurations, the mission planner realizes he can satisfy both requirements by selecting a payload configuration that alternates between the two configurations. This can only be done if the two requirements do not require constant monitoring of the areas of interest. The alternating between two configurations is possible because the satellite has two independent systems (visual and infrared) that share satellite resources (tape and power). By understanding the payload configurations, the mission plan-

ner can satisfy the lower priority requirement in addition to the higher priority requirement that is scheduled.

There are times when a lower priority requirement will be scheduled over a higher priority requirement due to payload configurations. Suppose there are three requirements with priorities 3, 4, and 5 (priority 3 is the highest and 5 the lowest) that occur during the same timeframe. If the priority 3 requirement is scheduled and the priority 4 requirement is not a compatible payload configuration, then the priority 4 requirement will not be scheduled during this timeframe. However, if the priority 5 requirement is compatible with the priority 3 requirement, then priority 5 will be scheduled with the priority 3 requirement during the specified timeframe. In this case, the priority 5 requirement would be scheduled, but not the priority 4 requirement.

Tape Recorders. There are two methods for receiving information from the satellite during operations. The first method consists of having the satellite transmit the data directly to a tracking station. This can be accomplished only when the station is operational and in view of the satellite. The second method consists of recording the information on a tape recorder and playing the data back when the satellite comes in view of a tracking station.

Two tape recorder constraints that the mission planner has to take into account are:

1. The number of cycles that are on a tape recorder.
2. The number of tape recorders onboard the satellite.

A tape cycle is the tape recorder operation that is composed of reading in data and playing it back. This operation is important because it is a mechanical operation and devices used in mechanical operations tend to be the first to fail. Therefore, mission planners can only use a limited number of cycles per day for each tape recorder to insure the tape recorders operate for the life expectancy of the satellite. Due to the limited number of tape cycles, the mission planner tries to use 100% of the tape recorder's capacity on every cycle to gather the maximum amount of data. This is why a mission planner attempts to fill one tape recorder before going to another one.

The number of tape recorders onboard the satellite also restricts the mission planner to the number of requirements that can be scheduled. Once a tape recorder is filled with data, the mission planner looks to see if another tape recorder is available. If there is, the mission planner continues to schedule more requirements. If there is not another tape recorder, the mission planner has to playback the data from the tape recorders at a tracking station prior to scheduling any more requirements.

As occurs in payload configurations, a lower priority requirement may be scheduled over a higher priority requirement but due to tape recorders instead of payload configurations. For example, suppose there are four requirements with priorities 1, 2, 3, and 4 (priority 1 is the highest and 4 the lowest) and the requirements with corresponding priorities 1, 2,

and 3 occur in sequence prior to a tracking station acquisition. The priority 4 requirement occurs after the same tracking station acquisition. The priorities 1 and 2 requirements use all of the available capacity of the tape recorders and the priority 3 requirement cannot be scheduled. Since no other acquisitions of the priority 3 requirement occurs after reading out the tape recorders at the tracking station, the priority 4 requirement is scheduled. Thus, the priority 4 requirement is scheduled instead of the priority 3 requirement because of tape recorder constraints.

Power. Power is the final satellite constraint. The power available on the satellite is based on three factors:

1. The type of satellite orbit.
2. The capacity of each satellite battery.
3. The number of batteries onboard the satellite.

The satellite orbit determines when the sun is shining on the solar arrays. Power is available via the solar arrays when the satellite is in the sunshine and via the batteries when the satellite is in the shadow of the earth. For a satellite in geosynchronous orbit or a satellite orbiting the earth, the satellite will be in the earth's shadow during certain times forcing operations to use power from the batteries.

When the mission planner is scheduling satellite operations, he must insure that critical satellite components remain on. These components include the satellite's communication and encryption subsystems which are required to communicate with the tracking stations. The available power for scheduling user

requirements is excess power which is not required to maintain these critical subsystems.

When an electronic component onboard the satellite is turned on, it uses power which is measured in ampere-hetoseconds (AHS). The mission planner keeps track of the components turned on and calculates the total AHS used during a desired payload configuration. If the power output from the solar arrays and the batteries is greater than or equal to the amount of power required for the critical subsystems and the desired payload, then the mission planner can schedule the desired configuration. If there is not enough power, the mission planner can modify the desired configuration to use less power or not schedule the requirement at all.

When the satellite is in the sunlight, plenty of power will be available for user requirements. However, when the satellite is in the shadow of the earth, the health of the satellite must be maintained by the batteries with any left over power available for user requirements. Thus, the power constraint is a function of the time the satellite is in the earth's shadow, the number of batteries onboard the satellite, and the capacity of each of the batteries. The mission planner can schedule a larger number of user requirements in the dark if more batteries are onboard the satellite and/or the capacity of each battery is increased.

Due to power, a lower priority requirement may be scheduled over a higher priority requirement similar to what occurred with the previous two satellite resources. For example,

suppose there are four requirements with priorities 1, 2, 3, and 5 (priority 1 is the highest and 5 the lowest) and the requirements with corresponding priorities 1, 2, and 3 occur in sequence in the earth's shadow. The priority 5 requirement occurs later in the orbit in the sunlight. The priority 3 requirement may not be scheduled because the satellite is in the dark and there is only enough power to schedule priorities 1 and 2 requirements. Later in the orbit, the satellite moves into the sunlight and the sun provides plenty of power to schedule the priority 5 requirement and other lower priority requirements. If no other acquisition of the priority 3 requirement occurs where power is available, then the priority 3 requirement will never be scheduled. Once again, a lower priority requirement is scheduled instead of the higher priority requirement.

Tracking Stations

In order to command and control the satellite, a communication link needs to be established. Tracking stations are the system segments on the earth that communicate with satellites. At the tracking station, commands are sent to the satellite to tell it what to do. Some of these commands are stored in the satellite's memory to control the satellite when it is out of view of the tracking station. In addition, the tape recorders are readout at the tracking stations to allow more data to be recorded.

Some satellite programs have dedicated tracking stations and support multiple satellites (such as the Defense Meteoro-

logical Satellite Program). Other satellite programs may have to share tracking station resources, such as the programs hosted at the Satellite Control Facility in Sunnyvale, California. In both cases, the mission planner must schedule the tracking station to insure no other satellite is using the station at the desired time.

When scheduling a tracking station, the mission planner analyzes the activities planned for the tracking station. For all station contacts, a certain amount of time is required to evaluate the status of the satellite and insure it is in good health. Once this is complete, the mission controllers (station personnel controlling the satellite) begin to command the satellite. The commands may include both reading out the tape recorders and/or loading commands for future operations. The mission planner must plan the tasks to be accomplished at the station and then selects a tracking station that has sufficient time to allow the mission controller to complete all the desired activities.

With these mission planning constraints established, the tasks that a mission planner must accomplish to generate a schedule can be described.

Mission Planning Tasks

The mission planner has many activities that must be accomplished to plan out a daily schedule for satellite operations. The following is a description of the activities that the mission planner performs throughout a day.

First, a mission planner must gather all the information required to schedule the satellite. This information includes: the active user requirements and their acquisitions, the available tracking stations and their acquisitions, and the available satellite resources. Once this information is gathered, the mission planner analysis the user requirements and the health of the satellite. The mission planner then schedules satellite operations using a prioritized list of requirement acquisitions. The acquisitions are assigned the priority of the corresponding requirement.

The scheduling of requirements is an iterative process. The flowchart in Figure 1 depicts the steps a mission planner performs. The mission planner starts at the highest priority acquisition and follows these steps:

1. Reviews the desired requirement's acquisition to verify that all the user constraints are met.
2. Checks to see if an acquisition is already scheduled during this time. If there is, he verifies the desired acquisition is compatible with the one already scheduled. If the acquisitions are not compatible, the mission planner notes the desired acquisition was not scheduled because it conflicted with a higher priority acquisition. If the desired acquisition was compatible or there was no other requirement was scheduled during this time, the mission planner attempts to select a tape recorder.

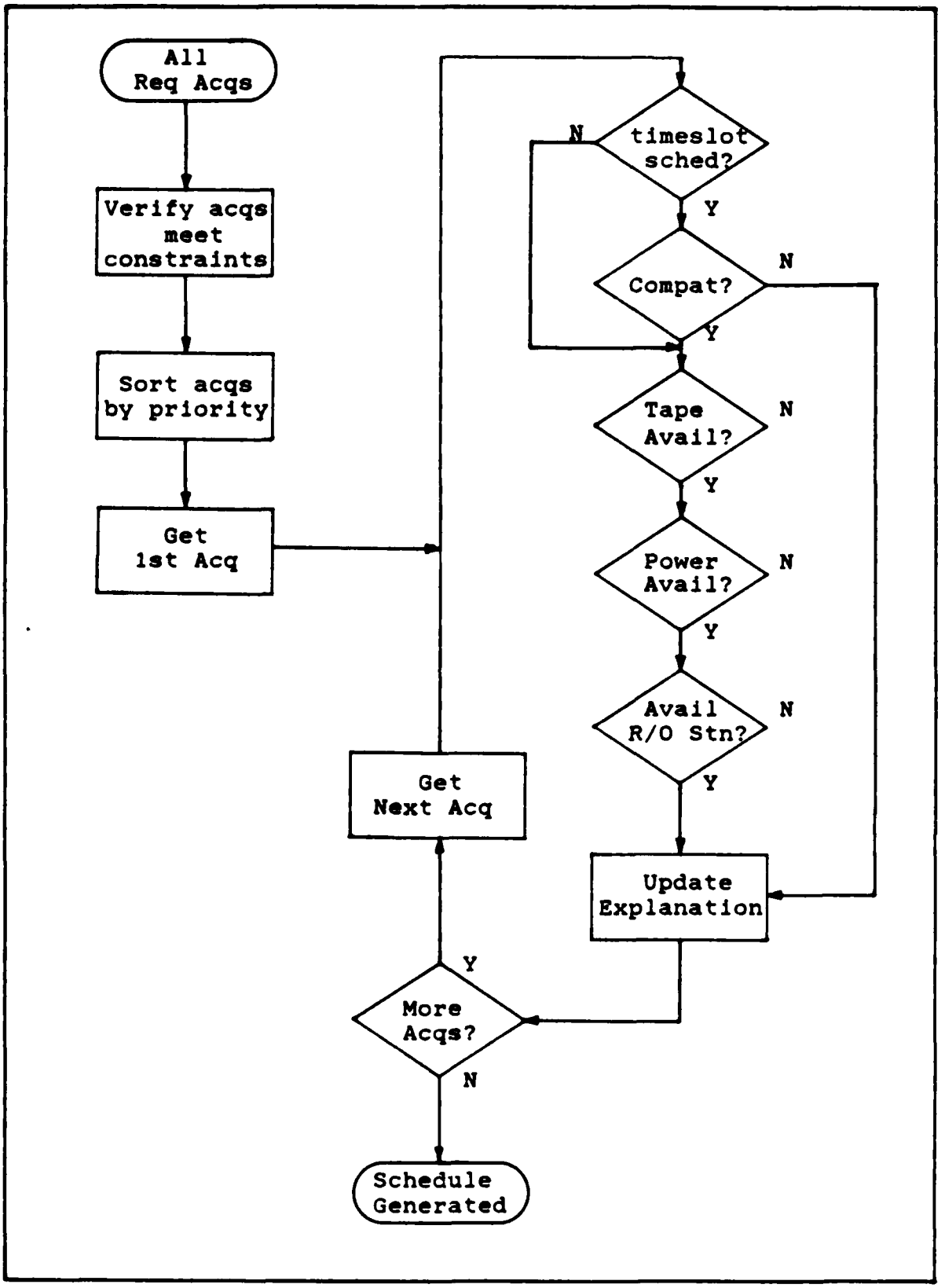


Figure 1. Scheduling Flowchart

3. Checks to see if tape is available on the tape recorder presently be used. If no tape is available, the mission planner looks to see if another tape recorder is available. If no other tape recorder is available then the mission planner attempts to readout the tape recorders at a station prior to the desired acquisition. If this is not possible, the mission planner notes the acquisition was not scheduled due to lack of tape. Otherwise, the mission planner proceeds to power.
4. Checks to see if power is available to read in and playback data from the acquisition. If there is no power, the mission planner notes the desired acquisition was not scheduled due to lack of power. Otherwise the mission planner attempts to schedule a tentative station.
5. Checks to see if a tracking station is available to readout the desired requirement. The station duration must be long enough to readout the tape and must occur after the desired acquisition. If such a station exists, the mission planner tentatively schedules it. If no station is available, the mission planner notes the desired acquisition was not scheduled due to lack of a readout station. In both cases, the mission planner proceeds to the next highest priority acquisition and starts with step 1.

As the mission planner generates the schedule, he may notice that a particular requirement was not scheduled. At this

point, the mission planner has the option of changing priorities and modifying the schedule or noting the desired changes and implementing them in the next schedule.

Summary

This chapter has presented an overview of satellite mission planning. The overall goal of the mission planner is to protect the health of the satellite while satisfying as many user requirements as possible. This is accomplished by performing the steps outlined in the Mission Planning Tasks section. Some of the limitations the mission planner has to consider are: constraints described in the user's requirements, orbital constraints, satellite constraints (payload configurations, tape recorders, and power), and tracking station constraints.

Now that the task domain has been presented, the system requirements can now be discussed.

IV. Requirements Definition

This chapter presents the requirements for a satellite mission planning knowledge-based system. First, hardware and software requirements are presented under the system requirements section. For the software requirements, differences in the operational system and the prototype system are reviewed. Next, the requirements for the knowledge-base are examined. Finally, the human-computer interface requirements are considered.

System Requirements

System requirements are separated into two categories: hardware and software. The hardware requirements are common to both an operational system and to the prototype system. Therefore, only one set of hardware requirements are discussed. Due to the time constraint associated with the thesis, the software requirements for the prototype system are a subset of the operational system. Hence, the operational software requirements are presented with the subset for the prototype specified.

Hardware. The type of hardware that a system is built on has to be carefully selected when the number of users that may be affected is large. Several organizations within multiple satellite programs may desire to use a mission planner advisor system so there will be a wide range of users. Therefore, easy dissemination of the prototype was a main consideration when

selecting an AT&T 6300 computer and an IBM AT. Most offices presently possess an IBM microcomputer (or compatible) thereby allowing the hardware to be an insignificant issue when acquiring a mission planning advisor system. In addition, the prototype would be highly transportable.

Software. The software requirements for both an operational system and the prototype system are based on defined heuristics, rules, and constraints obtained from a mission planner in addition to a single mission planning goal. The goal is to schedule all the user requirements by priority unless satellite or tracking station resources are not available or the desired user requirement conflicts with a requirement of higher priority.

The selection of a microcomputer tool/software package was the first step in generating the software requirements. Table II shows the different microcomputer tools/software packages available to the author with a list of the capabilities of each. Most of the systems examined in chapter 2 used frame-based knowledge representation. This observation supports the statement that frames are useful for problem domains where expectations about the form and content of the data play an important role in problem solving (27:74-75). Therefore, the tool selected had to have the capability to support frames, as described by Minsky (17), since scheduling satellite requirements is data intensive. This narrowed the choice to three: KES II, PC Plus, and PC Scheme. KES II and PC Plus do not support a frame system that contains frame templates and instan-

Table II. Knowledge-Based System Building Tools/Software Packages

Features	KES II	PC Plus	M.1	PC Scheme
Knowledge-Base Objects				
Frames	Yes	Yes	Must be programmed	Yes
Rules	Yes	Yes	Yes	Must be programmed
Certainty Factors	Yes	Yes	Yes	No
Inference Strategy (Backward Chaining)	Yes	Yes (also forward chaining)	Yes (also forward chaining)	Must be programmed
Interfaces				
Data Bases	Yes	Yes	Yes	Yes
External Programs	Yes	Yes	Yes	Yes
Sensors	Yes	No	No	No
Graphics Options	None	Being added	Being added	Limited

tiations of the template. These systems only support inheritance and permit procedures to search down a tree structure to find necessary parameters. In addition, KES II was on order and not available for development of the thesis prototype. Therefore, PC Scheme was selected as the software package to implement this prototype system.

The final step in generating software requirements was to determine the operational requirements that need to be coded into the system. This is where the requirements for the thesis prototype system became a subset of the operational system requirements. Table III shows the software requirements and the

Table III. System Requirements

Requirements	Thesis Prototype	Operational System
Generate a schedule based on priorities and resources	24 hr	24 hr
Graphics to display a satellite schedule	Sample 24 hr timeline only	24 hr
Graphics to zoom in on a window of the schedule and display what requirements compose the window	not implemented	any size window
Edit payload compatibility table for requirements	accomplished by modifying the code	able to modify
Add/Delete requirements	accomplished by hard coding the reqs	interactive
Edit requirements	edit all requirement slots	edit all requirement slots
Explanation capability	1) req was scheduled 2) why req was not scheduled	1) req was scheduled 2) why req was not scheduled 3) why req was modified
Schedules required remote tracking stations	has 2 RTS in database	has up to 16 RTS's in the database
Suggestions for scheduling requirements that were not previously scheduled	not implemented	required
Handles multiple satellites	not implemented	as many as the satellite program requires

Table III. System Requirements (cont)

Requirements	Thesis Prototype	Operational System
Edit satellite resources	available tape recorders, tape on specific recorders, tape cycles on specific recorders, and power	available tape recorders, tape on specific recorders, tape cycles on specific recorders, and power
Edit RTS resources	hard coded	when station is available and any limitations to the station
Has a power model	handles varying amounts of power during a given day	utilizes the power usage of each component on the satellite and the power generated from the batteries and solar arrays
Handles multiple satellite tape recorders	handles 2	handles the number on the satellite
Number of requirements handled	50	300
Generate a "what-if" capability	not implemented	the ability to make changes to satellite resources or requirements, generate a schedule, and determine if it is desirable
Generate requirements lists	1) priority ordered 2) time ordered	1) priority ordered 2) time ordered 3) by satellite

degree to which each requirement must be satisfied in both the operational and prototype systems. These requirements were

deleted from the prototype because of the amount of work and time required to implement the requirements and the limited time available to complete this thesis. These were determined not to impact the overall feasibility demonstration. Also some of the requirements only have to be partially implemented in order to demonstrate the feasibility of the knowledge-based system as done in the prototype. Extension of all these requirements to their full capability requires only time to implement. The coding and methods for these requirements are the same as those implemented. The final group of system requirements are those implemented in the prototype exactly as they would be in the operational system. These requirements are critical in showing the feasibility of using a knowledge-based system in satellite mission planning.

Knowledge-Base

The knowledge-base for the prototype knowledge-based system can be divided into three sections: user requirements, satellite constraints, and tracking station constraints. The knowledge was acquired by writing down cases the author experienced while working as a mission planner. All this knowledge is represented in the knowledge-base by frames.

The user requirements section must contain the requirement's name and the following information for each requirement: coordinates, start date, duration, priority, minimum acquisition duration, and desired payload configuration. The satellite constraints section must include the following information for each satellite: power available, tape available, number of

tape recorders, number of available tape cycles for each tape recorder, and payload configurations available on the satellite. Finally, the tracking station constraints section must consist of the following information for each of the tracking stations: tracking station's acquisitions, lock on time required for each tracking station, times the tracking station is not available, and satellite power required for each tracking station contact. Descriptions of all the previous items can be found in chapter 3, "Analysis of the Task Domain".

Human-Computer Interface

There is no concrete structured procedure or method to guide the design process with respect to the human-computer interface. However, in the thesis "Interactive Environment for a Computer-Aided Design System", Woffinden presents twelve design principles (31:40-52) that aid in designing a human-computer interface. Six of the principles strongly relate to this thesis and the system requirements. These principles are defined in the following subsections.

Determine the Purpose of the System. The purpose of the knowledge-based system is to assist the mission planner in scheduling satellite operations. This is accomplished by having the system gather all the information required to generate a schedule and then the system actually generates the schedule based on mission planner rules. After the schedule generation, each requirement contains an explanation specifically stating what the system did with the requirement. This

explanation includes the reason why a requirement was not scheduled or why it was modified. These explanations allow the mission planner to examine the requirements or database and make changes if desired. A knowledge-based system assisting the mission planner in this manner, allows the mission planner to concentrate on finding ways to schedule the largest number of requirements while optimizing the use of the limited resources.

Know the User. The designer of the prototype system served as a mission planning officer for over two years. He understands the duties of the mission planner and knows that most of the mission planners have no computer background. Therefore, the ability to input information and control the system should require minimum computer knowledge.

Identify Resources Available. Resources vary between satellite programs, but most of the program offices contain an IBM PC compatible computer. Therefore, the main resource to be used is the AT&T 6300.

Consider Human Factors. The user is assumed to have few, if any, handicaps and is able to operate a PC computer. However, the user is not a skilled typist so use of keyboard emphasizing typing skills will avoided. Consideration that the users have limited computer background indicates that a menu driven system should be implemented. In addition to menus, graphics such as satellite ground tracks and the satellite's

coverage of the earth would be helpful in an operational system.

Optimize Training. The system will be menu driven which allows a new user to do meaningful work without the assistance of an experienced mission planner.

Anticipate Errors. Since the users are not skilled typist, the system will verify that all the user's menu selections are valid inputs. If an error is found, the user is notified and a list of valid selections is shown.

Summary

This chapter presented the requirements definition for a knowledge-based system for satellite mission planning. The first section defined the system requirements in terms of the hardware and software requirements. The hardware requirements are the same for both the prototype and operational systems but the software requirements for the prototype system is a subset of the requirements for the operational system. These software requirements are summarized in Table III. The second section stated the information required in the knowledge-base. Finally, the third section defined the six principles that guide the design of the human-computer interface.

With the requirements having been defined for the system, the next chapter can describe how the requirements were implemented in the code.

V. Design

This chapter describes the design and implementation of the prototype knowledge-based system. The system is composed of user selections, input functions, output functions, the knowledge-base, the control strategy, and schedule functions. The main control of the system is performed by the user selections module. This module controls the operation of the prototype system by calling the other modules as the user dictates. These relationships can be seen in the system block diagram in Figure 2. The following sections summarize each of the modules and provide examples where possible.

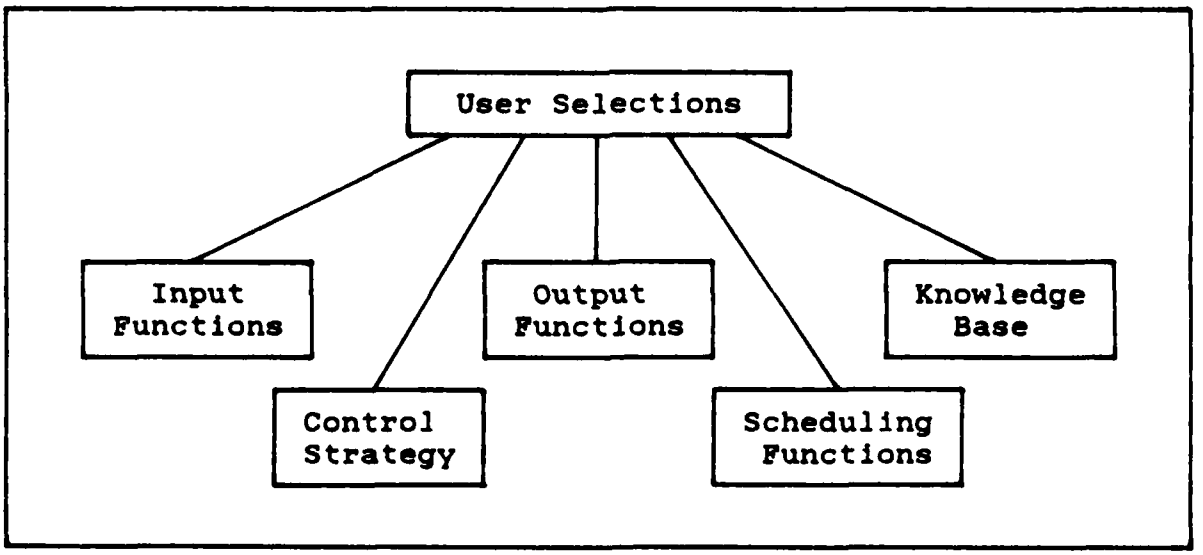


Figure 2. System Structure Chart

User Selections

The desires of the user are received by input options from two menus presented to the user. At each phase in the system's

operation, the system presents a menu to the user. The user then decides what option he desires and inputs the preferred number. This allows the user to guide and control the system's scheduling process. The main menu is shown in Figure 3.

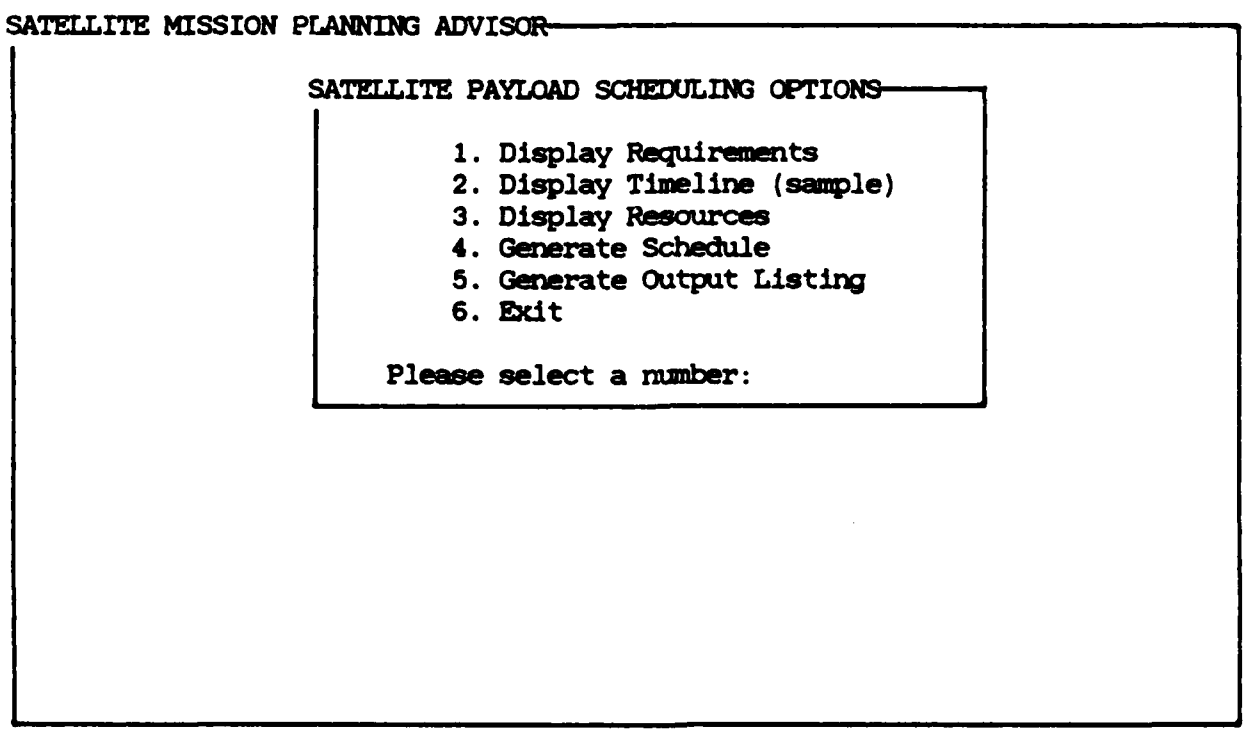


Figure 3. Main Menu Display

The menus were developed using the windows capability in PC Scheme. PC Scheme gives the programmer the options to specify window labels and border attributes in addition to declaring the size and location of the windows. The main menu window, "menu-window", was generated by the following code:

```
(define menu-window
  (make-window "SATELLITE PAYLOAD SCHEDULING OPTIONS" #!true))
(window-set-position! menu-window 3 20)
(window-set-size! menu-window 10 40)
(window-set-attribute! menu-window 'border-attributes 15)
```

Anytime the main menu is needed, the designation "menu-window" is used to indicate which window is requested. Using designations, the programmers can call desired windows anytime in the program without having to redefine the window.

Input Functions

The input functions were designed to input information through the files "resource.s", "reqs.db", and "variable.db". The file "resource.s" contains the initial values for the satellite power available, the available tracking stations, and the available tape cycles for tape recorders 1 and 2. The file "resources.s" looks like:

```
(define (load-initial-resources)
  (begin
    (send satellite1 set-initial-power
        '((0 800 790)(801 1230 775)(1231 1440 725)))
    (send tracking-station set-initial-tracking-station-acqs
        '((96 30 ts1)(219 30 ts2)(319 38 ts2)(443 33 ts1)
          (572 30 ts1)(669 30 ts2)(775 33 ts1)(888 28 ts1)
          (1026 26 ts2)(1158 40 ts2)(1265 23 ts1)
          (1397 29 ts2)))
    (send satellite1 set-initial-tape1-cycles 7)
    (send satellite1 set-initial-tape2-cycles 6)))
```

The file "reqs.db" contains all the user requirements. Each requirement is composed of the following information: requirement name, priority, payload mode, acquisition time, duration, and the power required per minute. An sample of the "reqs.db" file can be seen below:

```
req1    31    payload7    441    8    5
req2    64    payload9    417    12   3
req3    76    payload4    1298   13   3
req4    54    payload2    1390   11   5
req5    53    payload1    1259   8    5
req6    62    payload1    1408   1    1
req7    42    payload9    881    6    5
```

The information for each requirement must follow the given sequence since the software module reads in the values and assigns them to slots based on the assumption that the information is in the correct order.

The last file, "variable.db", defines all the variables which are used as designators for each instantiated requirement's class. This file presently has 50 variables defined and must be modified if the system is to increase the number of requirements that can be handled. A small segment of this file follows:

```
((define req0-1) (define req0-2)
 (define req0-3) (define req0-4
 (define req0-5) (define req0-6)
 (define req0-7) ..... (define req0-50))
```

The terms req0-1, req0-2, req0-3, etc. are arbitrary variables assigned to instantiations of the requirement's class unlike the terms req1, req2, req3, etc. in the previous file which are the names given to the requirements.

Making changes to the requirements (i.e. changing priorities) or modifying the available resources (i.e. the number of tape cycles) can be accomplished by editing the previously mentioned files. These files can be edited by any word processor but must be saved in ASCII text. This capability will not be required once an editor is designed in the knowledge-based system.

Output Functions

There are four output functions designed into the prototype system but only the first two are implemented in the

system. The first function allows the user to review the requirements on the screen both before and after the system generates a schedule. After a schedule is generated, the user can determine if the system scheduled a particular requirement and see an explanation of why the system performed the way it did. The user also has the capability to call two requirements up side by side to compare information. In addition, the system allows the user to examine the available tape and power resources before and after the schedule is generated. A sample display is show in Figure 4.

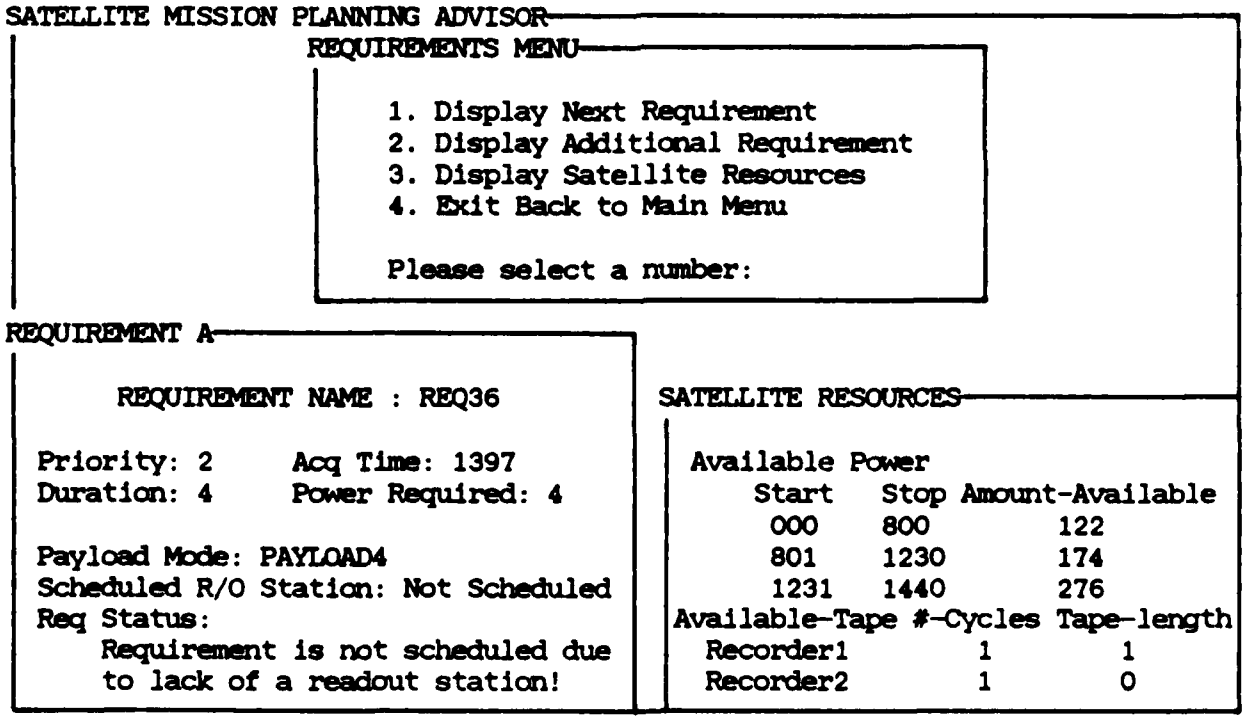


Figure 4. Knowledge-Based System Display

The second function allows the user to receive an output listing that summarizes the status of the requirements after

the schedule has been generated. Two types of listings are available: a priority ordered list and a time ordered list. The following shows the beginning and end of a priority ordered list:

NAME	PRIORITY	ACQ	DUR	END	POWER	PAYLOAD	STATUS	R/O-STN
REQ36	2	1397	4	1401	4	PAYLOAD4	no R/O stn	none
REQ26	2	1248	6	1254	3	PAYLOAD5	scheduled	1 1265
REQ33	3	619	14	633	3	PAYLOAD1	scheduled	2 669
REQ47	6	200	2	202	5	PAYLOAD7	scheduled	2 219
REQ24	10	374	3	377	1	PAYLOAD3	scheduled	1 443
:	:	:	:	:	:	:	:	:
:	:	:	:	:	:	:	:	:
:	:	:	:	:	:	:	:	:
REQ20	90	52	4	56	4	PAYLOAD2	no power	none
REQ18	91	974	8	982	2	PAYLOAD5	scheduled	2 1026
REQ8	92	1302	10	1312	5	PAYLOAD9	no tape	none
REQ17	95	906	3	909	2	PAYLOAD9	scheduled	2 1158
REQ28	95	1224	6	1230	2	PAYLOAD3	no tape	none

The time ordered list contains the same information but is sorted by time rather than priority.

The third function allows the user to graphically see a timeline of when the satellite is operating. A call for the timeline function is implemented but the function only produces a sample timeline which is shown in Figure 5. Since PC Scheme has very limited graphics capability, it was decided to use Turbo Pascal to draw the graphics. To generate an actual timeline, the schedule information is passed to Turbo Pascal by reading the timeslots into a file and then having them read out by Turbo Pascal. Once in Turbo Pascal, the timeslots are mapped into the timeline and correspond to every minute the satellite is operational.

The final function allows the user to save changes made to the requirements database. This function is implemented in the

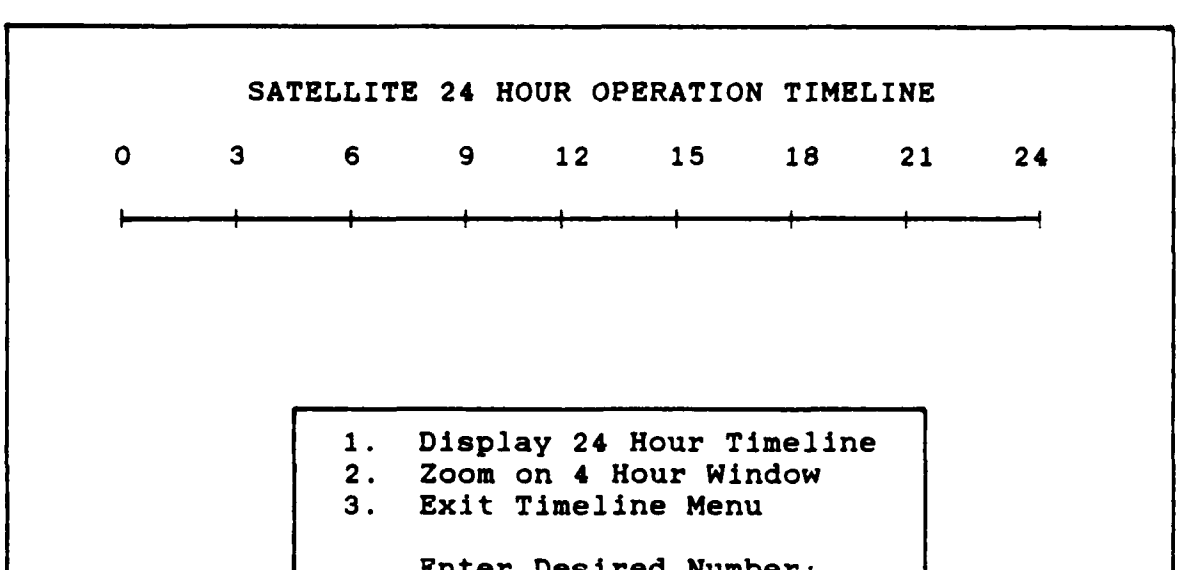


Figure 5. Sample Timeline

system by calling a dummy function. The capability to save database changes is designed to be used when an editor is built into the knowledge-based system. When changes are made to the database, the global variable *database-changes* is changed to yes and the user has the option to save the modified database.

Knowledge-Base

The knowledge-base is built in the Scheme Object-Oriented Programming System (SCOOPS) which allows frames to be implemented in the knowledge-base. The knowledge-base is divided into four classes: satellite resources, tracking station resources, requirements, and timeline slots. The knowledge-base and the classes that compose it are shown in Figure 6. Each class contains class variables and/or instantiation variables. In addition, each class can specify if the variables can be set, obtained, or initialized. A class variable is a variable

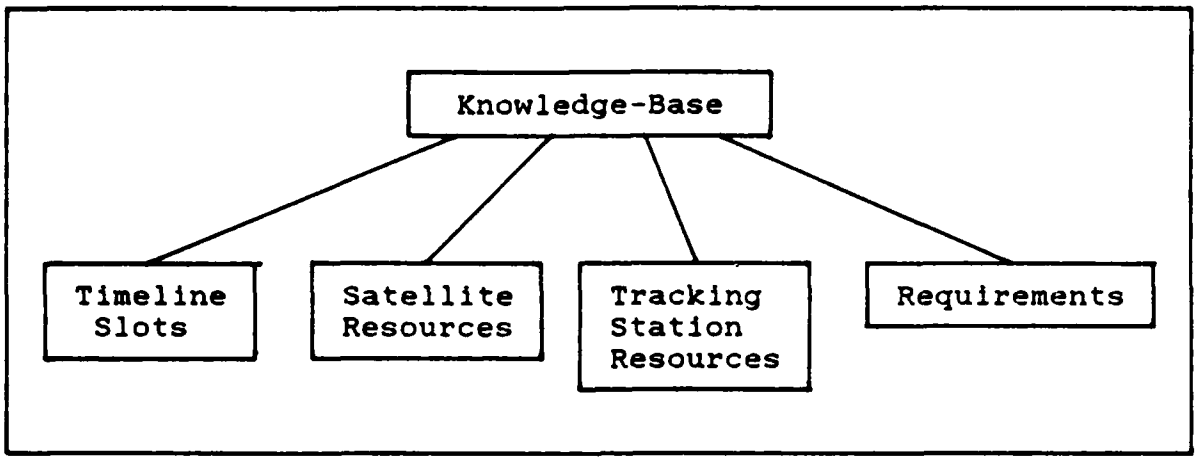


Figure 6. Knowledge-Base Structure Chart

that is common to all instantiations of the class. A instantiation variable is a variable that is associated with a specific instantiated class. Examples of a class variable and an instantiation variable can be seen in the following subsections. Information contained in instantiation variables is obtained and set by sending messages to the instantiated class. Examples are:

```
(send satellite1 get-available-power)
(send satellite1 set-available-power '124)
```

Methods can also be defined for a class which allows instantiation variables within the class to be obtained and/or set without sending messages. The method "update-scheduled-rts" which updates the class variable "scheduled-tracking-stations" in the class "timeline slots" is:

```
(define-method
  (tracking-station-resources update-scheduled-rts)
  (tracking-stn)
  (set-scheduled-tracking-stations
    (cons tracking-station scheduled-tracking-stations)))
```

Specific descriptions of each class is contained in the following subsections.

Satellite Resources. The satellite resource class contains information on the satellite's power, tape recorders, and payloads. The code that defines the satellite resources is:

```
(define-class satellite-resources
              (instvars
                (possible-payloads
                   '(payload1 payload2 payload3 payload4
                     payload5 payload6 payload7 payload8
                     payload9 payload10))
                initial-power
                initial-tape1-cycles
                initial-tape2-cycles
                (initial-tape-length 15)
                avail-power avail-tape1-cycles
                avail-tape2-cycles
                avail-tape1-length avail-tape2-length)
              (options
                gettable-variables settable-variables
                inittable-variables))
```

The instantiation variables in the satellite resource class is divided into two areas; initial variables which contain initial values for the parameters and available variables which contain the available values for the corresponding parameters. The instantiation variables (including a description) are:

1. possible-payloads: a list of possible satellite payloads
2. initial-power: a list of daily power initially available
3. initial-tape1-cycles: the daily number of tape cycles available for tape recorder 1
4. initial-tape2-cycles: the daily number of tape cycles available for tape recorder 2

5. initial-tape-length: the full duration of the tape recorders
6. avail-power: a list of power available after taking into account power required by scheduled requirements
7. avail-tape1-cycles: the number of cycles available on tape recorder 1 after subtracting cycles already used
8. avail-tape2-cycles: the number of cycles available on tape recorder2 after subtracting cycles already used
9. avail-tape1-length: the duration of tape on tape recorder 1 that is presently available
10. avail-tape2-length: the duration of tape on tape recorder 2 that is presently available

All values of initial variables (except "initial-tape-length") are set when the system is loaded. The values are read from the file "resource.s" during system initialization (refer to the input functions section to review the contents of "resource.s"). The values of the available variables are originally set to their corresponding initial variables and are updated as the system schedules requirements.

Tracking Station Resources. The tracking station resource class contains information on the tracking station's acquisitions, station lock on time, and the satellite power required to readout the tape recorders. The code that defines the tracking station resources is:

```
(define-class tracking-station-resources
              (instvars
                initial-tracking-station-acqs
                (lock-on-time 3) (power-reqd-per-minute 3)
```

```
    (scheduled-tracking-stations ())
    (tracking-station-acqs ()))
  (options
    gettable-variables settable-variables
    inittable-variables))
```

The instantiation variables within the tracking station resource class (including a description) are:

1. initial-tracking-station-acqs: a list of tracking station acquisitions which include the acquisition time, acquisition duration, and station name
2. lock-on-time: the time required for the station to acquire the satellite and verify the communication link
3. power-reqd-per-minute: the satellite power required per minute to readout the tape recorders
4. scheduled-tracking-stations: a list of scheduled tracking stations by acquisition time and station name
5. tracking-station-acqs: a list of available tracking station acquisitions that have been updated after a requirement is scheduled. Information includes the acquisition time, the remaining acquisition duration available, and the station name.

All the instantiation variables have initial values assigned except for "initial-tracking-station-acqs" and "tracking-station-acqs". The value for "initial-tracking-station-acqs" is read from the file "resource.s" during the system initialization. "Tracking-station-acqs" is originally assigned the value of "initial-tracking-station-acqs" and is updated as the system schedules requirements.

Requirements. The requirement class contains information on the user requirements. The code that defines the requirements is:

```
(define-class requirements
              (classvars list-of-requirements)
              (instvars
                requirement-name priority payload-mode
                acquisition-time duration
                power-required-per-time-unit
                explanations readout-station)
              (options
                gettable-variables settable-variables
                inittable-variables))
```

The instantiation variables within the requirements class are:

1. requirement-name: the requirement name
2. priority: a number between 1 and 100 indicating the requirement's relative importance with respect to the other requirements
3. payload-mode: the payload's required configuration
4. acquisition-time: the time the satellite acquires the requirement's location
5. duration: the length of time the satellite observes the requirement's location
6. power-required-per-time-unit: the power required per minute to read in the data in the desired payload configuration
7. explanations: contains the reasons why the system processed the requirement the way it did
8. readout-station: contains the station's name at which the scheduled requirement is readout

All values for each requirement are set during system initialization. All the variables (except 7 and 8) are read in from the file "reqs.db". When variables 1-6 are read in from the file, variables 7 and 8 are assigned default values (refer to input functions to review contents of "reqs.db"). When the system attempts to schedule the requirement, the explanation variable is updated to indicate the system's action. If the system schedules the requirement, the system also updates the readout-station variable The requirements class also contains a class variable. The class variable is "list-of-requirements" which contains a list of all the requirements in the system. When the requirements are initially loaded, the variable "list-of-requirements" is updated to include all the requirement names. Prior to scheduling, this list is sorted according to priority so the system attempts to schedule the highest priority first.

Timeline Slots. The timeline slots class contains schedule information. The code that defines the timeline slots is:

```
(define-class timeline-slots
              (classvars (list-of-timeslots ())
                         (list-of-timeslots-and-reqs ()))
              (options
                gettable-variables settable-variables
                inittable-variables))
```

This class contains two class variables: "list-of-timeslots" and "list-of-timeslots-and-reqs". The variable "list-of-timeslots" contains a list of all the timeslots that are scheduled. A timeslot corresponds to a specific minute of the day.

Use of timeslots is described in the scheduling functions section. The variable "list-of-timeslots-and-reqs" contains a list of the timeslots in addition to all the requirements that are scheduled during the specified timeslot. Both of these variables are initially set to nil and are updated as the system schedules the requirements.

Control Strategy

The control strategy used in the prototype was forward chaining with depth-first search. Due to the lack of an inference engine in PC Scheme, this strategy was embedded in COND statements. The statements were written to emulate the steps taken by a mission planner when scheduling the satellite. In addition to COND statements, both class methods and macros were used to control the knowledge-based system. Class methods are functions that have direct access to slots contained in an instantiated class and are used to process information in the specific class. The method "tape-check" demonstrates the use of methods and COND statements:

```
(define-method
  (requirements tape-check) (selected-requirement)
  (write-char #\return scheduling-window)
  (write-char #\tab scheduling-window)
  (princ "doing tape check" scheduling-window)
  (set! *selected-recorder-cycles* ())
  (cond
    ((and (>=? (send satellite1 get-avail-tape1-length)
               duration)
          (>=? (- (send satellite1 get-avail-tape1-length)
                  duration) 0))
     (begin
       (set! *selected-recorder* 'recorder1)
       (send-requirement-for-power-allocation
         (eval *selected-requirement*))))
```

```
((and (>=? (send satellite1 get-avail-tape2-length)
           duration)
      (>=? (- (send satellite1 get-avail-tape2-length)
              duration) 0))
 (begin
   (set! *selected-recorder* 'recorder2)
   (send-requirement-for-power-allocation
     (eval *selected-requirement*))))
((>? (send satellite1 get-avail-tape1-cycles) 1)
 (begin
   (set! *selected-recorder* 'recorder1)
   (set! *selected-recorder-cycles* 'recorder1)
   (send-requirement-for-power-allocation
     (eval *selected-requirement*))))
((>? (send satellite1 get-avail-tape2-cycles) 1)
 (begin
   (set! *selected-recorder* 'recorder2)
   (set! *selected-recorder-cycles* 'recorder2)
   (send-requirement-for-power-allocation
     (eval *selected-requirement*))))
(else
  (send-not-scheduled-due-to-tape
      (eval *selected-requirement*)))))
```

Macros are used to allow procedures and functions access to multiple instantiated classes. When trying to access information in SCOOPS (the frame-based system in PC Scheme), procedures and functions could not be evaluated directly. Therefore, macros were used. Macros use one or more argument expressions to build an intermediate form and then the intermediate form is evaluated to produce an output value. "Get-req-name" is an example of a macro:

```
(macro get-req-name
  (lambda (e)
     (list 'send (cadr e) 'get-requirement-name)))
```

Scheduling Functions

All the functions and procedures used for scheduling user requirements check to see if a requirement is compatible, tape is available, power is available, and there is a readout sta-

tion available. The method "tape-check", which is displayed in the previous section, is an example of a function that is used to schedule user requirements. If the requirement satisfies the all the checks previously mentioned, the software module allocates resources to the requirement. Once the module has allocated resources, the timeline slot class is updated to correctly reflect the schedule. If the requirement does not satisfy any of the checks, the software module updates the requirement's explanation slot.

The timeline slots class is the focal point for schedule information. The timeline slots class contains the class variables "list-of-timeslots" and "list-of-timeslots-and-reqs" (as shown in the subsection "Timeline Slots"). When the system initially starts to schedule user requirements, the two class variables are set to nil. As the system determines a requirement can be scheduled, timeslots are created. A timeslot is a number that corresponds to a specific minute the requirement is scheduled. For every minute that a requirement is scheduled, a corresponding timeslot is generated. If a timeslot already appears in the "list-of-timeslots", the system simply adds the requirement name to the corresponding timeslot listed in the "list-of-timeslots-and-reqs". If the timeslot does not appear in the "list-of-timeslots", the system adds the timeslot to the "list-of-timeslots" and adds the timeslot and corresponding requirement name to the "list-of-timeslots-and-reqs".

This process is carried on as long as the system is scheduling requirements. After the scheduling process is complete,

the timeslots contained in the "list-of-timeslots" will correspond to every minute the satellite is operational.

With the design of the prototype system presented, the system can now be analyzed and evaluated to determine if the system requirements were met.

VI. Analysis and Evaluation

This chapter presents the analysis and evaluation of the prototype knowledge-based system. The system was evaluated in three separate areas: the prototype system, the software environment, and the hardware environment. Each area is analyzed and evaluated in the following sections.

Prototype System

The prototype system was considered working if it scheduled user requirements based on priorities and resources available. If a requirement was not scheduled then its explanation would state that it conflicted with a higher priority requirement or a particular resource was not available. Two methods were use to verify the system's operation; evaluation of the scheduling process using six cases containing only two requirements and evaluation of the scheduling process using three cases containing 50 randomly generated requirements. Table IV summarizes the nine cases and indicates the specific software area that each case tested and verified. These areas are main components of the scheduling functions module and are critical to the systems operation. Since the scheduling functions module is decomposed into specific areas, each area can then be tested and validated by only a two requirement case. In cases 3, 4, and 5 several areas were not tested because the system immediately exits the scheduling process when one of the areas cannot be satisfied. Each of these methods are reviewed in more detail in the following subsections.

Table IV. System Test Matrix

Software Areas Tested and Validated	Cases								
	1	2	3	4	5	6	7	8	9
Priorities taken into account	X	X	X	X	X	X	X	X	X
Compatibility check	X	X	X	X	X	X	X	X	X
Tape recorder check	X	X		X	X	X	X	X	X
Power check	X	X		X		X	X	X	X
Readout station check	X	X				X	X	X	X
Resource allocation	X	X					X	X	X

Two Requirement Cases. The system was evaluated against the two requirement cases because the process of scheduling satellite operations is composed of individual processes of scheduling each requirement. This allowed the system to be evaluated using a simplified requirements database which simplifies the testing procedure. There were 6 two requirement cases that the system was tested against. The six cases were:

1. Non-overlapping requirements with plenty of resources
2. Overlapping compatible requirements with plenty of resources
3. Overlapping incompatible requirements with plenty of resources
4. Non-overlapping requirements with insufficient power
5. Non-overlapping requirements with insufficient tape

6. Non-overlapping requirements with no station to readout the tape recorders

After a minor modification to the program, the system performed well in all six cases. As time allowed, several variations of these cases were tested. These variations also indicated the system was running according to system requirements.

Randomly Generated Requirements Cases. After the initial test with the six simplified cases, the system was tested three more times using a requirements database with 50 randomly generated requirements. The randomness of the requirements in Table V. A sample of one of these cases can be found on

Table V. Range of Random Requirements

Requirement Segment	Range
priority	1 - 100
payload configurations	1 - 10
start time	0 - 1410
duration	1 - 15
required power/minute	1 - 5

page 40. Again, the program needed several minor modifications in order to perform correctly. After the changes, the system performance for all three cases supported the fact that the prototype system ran according to the system requirements.

Software Environment

The software environment of PC Scheme generally supported the prototype system well. However, there were two problems that occurred. The first problem, a major one, was uncovered while the code was being written to read the requirements from the file "reqs.db". A variable has to be assigned to each user requirement in the database and the number of requirements is not necessarily known. This variable is used to access the frame of each requirement. As requirements are read in, the system should create a variable and assign it to the requirement. The problem is that this cannot be done in PC Scheme. In Common Lisp this is simple because the function 'setq can be used to define a variable and set it equal to an item. However, there is no 'setq in PC Scheme. In fact, the only way to define a variable is to use the function 'define. The problem is that you cannot have a 'define inside a defined function. One option was to use macros to use 'define in specific functions, but any other function calling the function containing the macro also has to be called by a macro. This was not acceptable because the majority of the input module would have to be written using macros. Instead, the file "variable.db" was created to define all the variables necessary to assign the requirements to specific frames. This solution requires the file "variable.db" to be modified any time the number of user requirements increases to over 50. The problem was brought to the attention of technical representatives at Texas Instruments and they could not find a way to resolve it.

The second problem that occurred was due to the lack of an inference engine in PC Scheme. This meant that every control rule had to be explicitly coded into the program. The prototype system was simplified enough that this caused only minor problems. However, if the prototype is going to be extended to an operational system, an inference engine needs to be added.

Hardware Environment

The hardware environment never showed any problems with running the prototype system. However, there was a problem that was encountered when constructing the system. The system could not support both the Scheme environment and the Edwin environment (the LISP editor) at the same time. Due to the limited RAM memory of the microcomputer (640K), programs quickly grew to a size that caused the system would go into an infinite loop when trying to compile a file in edwin and then test it in the scheme environment. A work around was devised by exiting PC Scheme and using a word processing editor. However, this negated the advantages of using the edwin environment (i.e. matching parentheses, evaluating programs without loading them, etc.).

Another potential area of concern is the time required to do garbage collecting. The system time spent garbage collecting increased when the number of requirements in the database increased from 2 to 50. This is a potential problem because if 300 requirements are in the database, it is questionable if the system could handle the computations with the limited memory of a microcomputer. Testing the system with small increments in

the number of requirements will be the only way to evaluate if the limited memory will be a problem.

With the assessment of the prototype system's operations complete, the conclusions and recommendations can now be presented.

VII. Conclusions and Recommendations

This chapter presents a synopsis of the thesis project. This is accomplished by first summarizing the thesis project and presenting conclusions based on the project. Next, improvements and enhancements to the system are suggested. Finally, future areas where this thesis project could be extended are discussed.

Summary

The purpose of this thesis was to take an initial look at the feasibility of using a knowledge-based system to assist mission planners in satellite operations. A prototype knowledge-based system was built on an IBM compatible computer which showed that the idea of a knowledge-based system for satellite mission planning was indeed feasible. However, the system has several difficulties that will need to be overcome in order to have operational system. First, the system must be upgraded to meet the operational requirements and not just the subset the prototype system was designed to implement. Second, the software environment that the prototype was designed in was marginally acceptable. The problem encountered with defining variables required by the system must be resolved if the knowledge-based system is to become operational. As the system receives new user requirements, it must be able to create new variables for the required frames without having to have them predefined. Having variables predefined limits the number of user requirements that can be handled and is an

unnecessary constraint. In addition, consideration has to be given to finding a new software package that not only contains a frame-based knowledge representation system but also an inference engine. An inference engine is required to allow the user, knowledge engineer, and programmer more flexibility in constructing the knowledge-based system. As the system becomes more complex, the user, knowledge engineer, and programmer will be able to produce a system without explicitly specifying every rule required. An inference engine could be added to PC Scheme, however, the hardware environment may not support the expansion required for an operational system. This leads to the final area of concern, the hardware environment. Designing this knowledge-based system on an IBM compatible computer has many advantages but this type of knowledge-based system is a too advanced for the present state of microcomputers. RAM memory limitations pose the biggest problem for knowledge-based systems using microcomputers. Microcomputers are being upgraded to several megabytes of memory but not all utility programs are able to access the full amount of available memory. When it is a common for microcomputers to have several megabytes of RAM memory and all utility programs access the full amount of RAM, then microcomputers will be ready and capable of supporting large operational knowledge-based systems.

Recommended Improvements

There are several areas where improvements can be made. These improvements can be divided into two areas, the way the

system is implemented and the capabilities of the system. The suggested changes with the way the system is implemented are:

1. Consider upgrading the system from PC Scheme to PC Plus after PC Plus is upgraded. Presently, PC Plus is a rule-based system with an inference engine and allows inheritance of rules in a tree structure. Texas Instruments is presently working on an upgrade that would furnish a frame-based knowledge representation capability within PC Plus. This would give the capability of both rules and frames with an inference engine on a microcomputer.
2. If the upgraded version of PC Plus does not work, then I suggest looking into moving to a Lisp machine. The way prices are dropping on Lisp machines makes them a viable alternative to the minicomputer. In addition there are many knowledge-based system tools such as FLAVORS, LOOPS, and KEY already available.

There are many improvements that must be made to the prototype system to make it operational. However, there are three that are recommended to upgrade the prototype to the next level of development. The three recommended improvements are:

1. To have the system schedule the acquisitions of user requirements and take into account when a requirement has been satisfied. A requirement usually has multiple acquisitions, depending on the satellite's orbit, and when only one acquisition must be satisfied the system

should schedule other requirement acquisitions even though they may be of lower priority.

2. Creating the capability for the knowledge-based system to handle soft and hard constraints for each requirement. A user requirement usually has constraints that must be met in order to satisfy the requirement. These constraints are defined as hard constraints. There are other constraints the user desires but do not have to be met in order for the requirement to be satisfied. These constraints are soft constraints. The capability to differentiate between these constraints provides the system greater flexibility in generating the schedule.
3. Improve the explanation capability of the knowledge-based system. Not only should the system give the reasons for the way it scheduled the requirement, but when the requirement was not scheduled, the system should also provide suggestions for changing requirements so that it can be scheduled. Each suggestion should also state what other requirements may be impacted if this suggestion is implemented.

Future Extensions

If the results of the prototype continue to prove as positive as the initial examination indicates, there are two areas where application of this prototype may apply.

The first area would be to tailor the prototype to a specific satellite program. This would include creating a

knowledge-based system interface to software modules that generate satellite ephemeris and tracking station acquisitions. In addition, all the capabilities of the specific satellite would have to be incorporated into the knowledge-based system. This would include information about the tracking stations that support the specific satellite.

The second area for extension would be to use this knowledge-based system for autonomous on-orbit satellite mission planning. If this system proved to generate viable schedules, then the system could generate a mission plan onboard the satellite as a contingency for satellite operations. For example, assume the satellite has a timer onboard. If after a certain amount of time has passed with no tracking station contact, the satellite will begin to generate its own mission plan and automatically transmit the data when it came into view of the tracking stations. The information required onboard the satellite would be the user requirements, the satellite's ephemeris, constraints on the satellite's resources, and the tracking station locations.

Conclusion

This thesis effort was designed to take an initial look at the feasibility of using a knowledge-based system to assist mission planners in satellite operations. A prototype knowledge-based system was built on an IBM compatible computer and showed that the idea of a knowledge-based system for satellite mission planning was indeed feasible. Several upgrades and improvements have to be made to the prototype system to make it

operational, but this thesis has provided a foundation for the development of an operational mission planner knowledge-based system.

Bibliography

1. Abelson, Harold, et al. Structure and Interpretation of Computer Programs. New York: McGraw-Hill Book Company, 1985.

2. Bahnij, Robert B. A Fighter Pilot's Intelligent Aide For Tactical Mission Planning. MS thesis. Wright Patterson AFB, Ohio: School of Engineering, Air Force of Technology, December 1985.

3. Barr, Avron. and Edward A. Feigenbaun. The Handbook of Artificial Intelligence, Volume 2. Los Altos, California: William Kaufman, Inc., 1981b.

4. Barr, Avron. and Edward A. Feigenbaun. The Handbook of Artificial Intelligence, Volume 1. Los Altos, California: William Kaufman, Inc., 1981a.

5. Charniak, Eugene, et al. Artificial Intelligence Programming. Hilldale, New Jersey: Lawrence Erlbaum Associates, Inc., 1980.

6. Cohen, Paul R. and Edward A. Feigenbaun. The Handbook of Artificial Intelligence, Volume 3. Los Altos, California: William Kaufmann, Inc., 1982.

7. Draper, Stephan W. and Donald A. Norman "Software Engineering for User Interfaces," IEEE Transactions on Software Engineering, SE-11: 252-258 (March 1985).

8. Evans, David D. and Major Ralph R. Gajewski, USAF. "Expanding Role for Autonomy in Military Space," Aerospace America, 74-77 (February 1985).

9. Fox, Mark S. Constraint Directed Search: A Case Study of Job-Shop Scheduling. PhD dissertation. Computer Science Department, Carnegie-Mellon University, Pittsburgh Pa, 1983.

10. Goldstein, I.P. and R.B. Robert. "Using Frames in Scheduling," Artificial Intelligence: An MIT Perspective Volume 1. Cambridge, Mass.: The MIT Press, 1979.

11. Grenander, Sven. "Toward the Fully Capable AI Space Mission Planner," Aerospace America, 44-46 (August 1985).

12. Harmon, Paul and David King. Expert Systems: Artificial Intelligence in Business. New York: John Wiley and Sons, Inc., 1985.

13. Hayes-Roth, Frederick, et al. Building Expert Systems. Reading, Massachusetts: Addison-Wesley Publishing, 1983.

14. Hayes-Roth, Frederick, et al. "The Knowledge-Based Expert System: A Tutorial," Computer, 17: 11-28 (September 1984)b.

15. Hayes-Roth, Frederick, et al. "Knowledge-Based Expert Systems," Computer, 17: 263-273 (October 1984)a.

16. Koch, Fred H., MS Student. Summary of Scheduling Expert Systems in Artificial Intelligence generated for OPER699, Independent Study Course. School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, Ohio, 1986.

17. Minsky, M. "A Framework for Representing Knowledge," The Psychology of Computer Vision. New York: McGraw-Hill, 1975.

18. Nachtsheim, P.R., et al. A Knowledge-Based Expert System for Scheduling of Airborne Astronomical Observations. Technical Memorandum 88194. Moffet Field, Ca.: Ames Research Center, NASA, December 1985.

19. Parnell, Gregory S. "Artificial Intelligence and Space Operations: Implications for Aerospace Doctrine," Tenth Air University Aerospace Power Symposium, Maxwell Air Force Base, Alabama: Air War College, 1986.

20. Rich, Elaine. Artificial Intelligence. New York: McGraw-Hill Book Company, 1983.

21. Robinson, Ann and David Wilkins Man-Machine Cooperation for Action Planning. SRI International. Menlo Park, Ca., 1982. (AD-A 124 243)

22. Semeco, Antonio C., et al. "GENSCHED - A Real World Hierarchical Planning Knowledge-Based System," SPIE Applications of Artificial Intelligence III, 635: 250-256 (1986).

23. Slagle, James R. and Henry Hamburger. "An Expert System for a Resource Allocation Problem," Communication of the ACM, Volume 28, Number 9: 994-1004 (September 1985).

24. Srivastava, Dr. Sadananad. Autonomous Scheduling Technology for Earth Orbital Missions. NASA-CR-168939; Bowie State College, Maryland: Goddard Space Flight Center, NASA, 1982. (N82-29217)

25. Troussaint, A.L. and M.E. McFall. "Spacecraft Application of Expert Systems," Proceedings of the IEEE, National Aerospace and Electronics Conference: 1342-1346 (May 1985).

26. Vere, Steven. "Deviser: An AI Planner for Spacecraft Operations," Aerospace America, 50-53 (April 1985).

27. Waterman, Donald A. A Guide to Expert Systems. Reading, Massachusetts: Addison-Wesley Publishing Company, 1986.

28. Wilensky, Robert. LISPcraft. New York: W.W. Norton and Company, 1984.

29. Winston, Patrick Henry. Artificial Intelligence (Second Edition). Reading, Massachusetts: Addison-Wesley Publishing Company, 1984.

30. Winston, Patrick Henry. and Berthold Klaus Paul Horn. Lisp (Second Edition). Reading, Massachusetts: Addison-Wesley Publishing Company, 1984.

31. Woffinden, Duard S. Interactive Environment for a Computer-Aided Design System. MS thesis. Monterey, California: Naval Post-Graduate School, June 1984.

Appendix A

Prototype User's Guide

Getting Started

The knowledge-based system for satellite mission planning is designed so the user of the system needs no experience with a computer. However, several files are required in order to have the system operate. These files and a description for each are listed below:

1. thesis.fsl - the source code for the prototype expert system
2. reqs.db - contains all the user requirements which the system will attempt to schedule
3. resource.s - contains the initial values for the satellite power, tracking station acquisitions, and available tape cycles
4. variable.db - contains all the variables that are used to assign the requirements to frames

The system can be initialized by two methods: automatic initialization and manual initialization. Each of these methods are described in the following subsections.

Automatic Initialization. A person with minimum computer experience can modify the file "scheme.ini" for the prototype system to automatically be initialized when the user calls up the PC Scheme environment. Using an editor that can save a file as an ASCII "DOS" text file, a person should enter the following statements: (load "thesis.fsl") and (ss). These

statements will load the source code for the prototype system, load the requirements, and present the main menu to the user.

Manual Initialization. The user can initialize the system manually if the file "scheme.ini" has not been modified to include automatic initialization. The user first loads the PC Scheme environment into the computer system. When the cursor prompt appears, the user types the statement '(load "thesis.fsl")'. This causes the source code for the prototype system to be loaded. When the cursor prompt for PC Scheme appears again, the user types in the statement '(ss)'. This causes the system to load the user requirements and then presents the user with the system's main menu.

System's Operation

Menus were used to give the user control over the operation of the system. At each menu, the system requests the user to select an option by inputting the number corresponding to the desired selection. Once the number is entered, the system verifies the input is a valid selection. If the input is not valid, the error window appears and states the input was invalid. In addition, the error window presents valid options and requests the user to make another selection. If the input is valid, the system proceeds to implement the desired option.

Two options need to be mentioned. First, if a user desires to generate output listings, the listings will be found in the file "time.lst" for the time ordered list and in the file "priority.lst" for the priority ordered list. When the

listings are generated, any information previously in the files will be lost. Second, if the user desires to exit the system, the user must select the exit option at the main menu. The remainder of the options are self explanatory.

Modifying Database Information

Having the capability to modify database information allows the user to investigate other alternatives that may allow the mission planner to satisfy more requirements. Although the prototype system does not have a built in editor that allows modification to database information, the system was designed so the user can make changes to the database by using any editor that can save files as ASCII "DOS" text files. Two files contain all the data information that needs to be modified (a description of these two files can be found in Chapter 5 on page 40). The first file, "reqs.db", contains all the requirements which the system will attempt to schedule. If the user wants to change any parameters of a specific requirement, he simply modifies the requirement in the file and saves it as an ASCII "DOS" text file. The second file, "resource.s", contains the initial information that is used by the system. The information that can be modified in the file is: the satellite power available, the tracking station acquisitions, and the number of available tape cycles.

System Constraints

The system has several constraints that limit the capabilities of the prototype system. The constraints are:

1. The number of variables - Presently, the system is designed with 50 variables in the file "variable.db". The variables are used to assign requirements to frames. If more than 50 requirements are desired to be scheduled, then "variable.db" has to be modified to generate as many variables as there will be requirements.
2. The number of tape recorders - The system is designed to schedule requirements using only two recorders on the satellite.
3. The power module - The system has the day divided into three time spans and checks the time span the requirement occurs in to verify that there is enough power to schedule the requirement. The system does not take into account the power used by each satellite component or the power generated by the batteries and the solar arrays.
4. The number of satellites - The system is designed to schedule only one satellite per set of requirements.

Appendix B

Operational Concept Using a Knowledge-Based System

There are many people that participate in satellite mission planning and scheduling. Table A-I summarizes the tasks that are performed by the different support personnel. The knowledge-based system is designed to directly assist the satellite scheduler in scheduling satellite operations. The system performs the satellite scheduler's actions thus allowing the scheduler to concentrate on ways to schedule more user requirements while optimizing the available resource. In addition to assisting the satellite scheduler, the knowledge-based system could assist the satellite users and mission planners and perform onboard satellite operations. The following three sections discuss how a system could assist satellite users, mission planners, and conduct onboard mission planning.

User's Implementation

A knowledge-based system could assist a satellite user by allowing him to make requirement changes (i.e. change priorities, add requirements, delete requirements, etc.) and then permit him to check the effects of the changes prior to implementation. This allows the user the opportunity to optimize the requirements and the requirement's priorities prior to passing them on for implementation.

Mission Planner's Implementation

A knowledge-based system could assist a satellite mission planner by generating an initial schedule and then allowing the

Table B-I. Satellite Mission Planning and Scheduling

Legend
- A - Satellite Users
- B - Mission Planners (System Program Office)
- C - Satellite Engineers (System Program Office)
- D - Satellite Schedulers (System Program Office or Mission Control Complex)
- E - Mission Control Complex Personnel
- F - Range Operations Schedulers

Satellite Mission Planning and Scheduling Tasks	Support People					
	A	B	C	D	E	F
1. Issues requirements	X					
2. Analyzes & combines user & health requirements to determine best use of satellite		X				
3. Generates tentative schedule for individual satellite programs				X		
4. Schedules mission control complex resources/identifies tracking station needs					X	
5. Notification of conflicts/allocation of ground support systems						X
6. Reschedules mission control complex resources to meet allocated ground support systems					X	
7. Commands satellite					X	
8. Maintains health of satellite		X	X	X	X	
9. Analyzes satellite health telemetry			X			
10. Analyzes satellite payload data	X					

Notes: "X" indicates the specified task is done by the specified support person.

mission planner to go back and find out what requirements did/did not get scheduled. The system would also provide the reasons why a requirement did not get scheduled and would recommend changes that would allow the requirement to be scheduled. The mission planner could then make the changes and have the knowledge-based system generate an schedule with the new changes. This allows the mission planner to observe the impact of the satellite engineering constraints on the number of requirements that can be satisfied. The mission planner can use this information to determine if the engineering constraints are so conservative that the mission of the satellite can not be accomplished.

Onboard Satellite Mission Planning

A knowledge-based system designed to assist in scheduling satellite operations could be used to do simplified onboard mission planning during contingency operations. The satellite would be programmed to plan and execute its own satellite operations if it is not contacted after a specific amount of time. The satellite would maintain the status of all its components, its orbit, the location of tracking stations, and the power available. With this information, the satellite would attempt to satisfy any mission requirements that were previously loaded. This allows satellite operations to occur in any contingency condition.

The growing complexity of modern satellites and limited resources available for satellite operations has caused satellite mission planning to become a data intensive job which overwhelms mission planners. The purpose of this research was to determine the feasibility of using Artificial Intelligence techniques, specifically knowledge-based systems, in satellite mission planning.

Research was conducted to determine the type of knowledge representation that best accommodates data intensive problem domains. Using this basis, a prototype knowledge-based system for use on a microcomputer was designed, constructed, and evaluated. The satellite schedule was generated based on prioritized user requirements, the requirements' acquisitions, and available resources. Explanations were provided to enable the mission planner to understand how the schedule was generated and allow him to make changes as user requirements change.

The prototype system showed that a knowledge-based system can assist the mission planner in scheduling satellite operations. This establishes a base from which more research can be done to help determine the optimal environment required to support an operational mission planning knowledge-based system.

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

A178893

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) AFIT/GE/ENG/86I -17	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION School of Engineering 6b. OFFICE SYMBOL (If applicable) AFIT/ENG	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State, and ZIP Code) Air Force Institute of Technology Wright-Patterson AFB, Ohio 45433	7b. ADDRESS (City, State, and ZIP Code)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Satellite Control Facility 8b. OFFICE SYMBOL (If applicable) AFSCF/DVE	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code) Sunnyvale, Ca 94086	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO / WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)

A PROTOTYPE KNOWLEDGE-BASED SYSTEM FOR SATELLITE MISSION PLANNING (U)

12. PERSONAL AUTHOR(S)
Perales, David E., B.E.E., Capt, USAF

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
MS Thesis	FROM _____ TO _____	1986 December	87

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Artificial Intelligence Artificial Satellites
06	04		

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

(see reverse)

Approved for public release: IAW AFR 190-1.
LYNN E. WOLAVER
Dean for Research and Professional Development
Air Force Institute of Technology (AU)
Wright-Patterson AFB OH 45433

20 DISTRIBUTION/AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	21 ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a NAME OF RESPONSIBLE INDIVIDUAL Gregory Farnell, LtCol, USAF	22b TELEPHONE (Include Area Code) 513-255-3362
	22c OFFICE SYMBOL AFIT/ENS

DD Form 1473, JUN 86 — Previous editions are obsolete — SECURITY CLASSIFICATION OF THIS PAGE

UNCLASSIFIED

Vita

Capt David E. Perales was born to Heriberto and Juanita Perales on 14 August 1957 in Selma, Alabama. He graduated from Salpointe High School in Tucson, Arizona in 1975 and attended Auburn University, Auburn, Alabama from which he received the degree of Bachelor of Electrical Engineering in August 1980. Upon graduation, he received a commission into the USAF through the ROTC program. In October 1980, he was assigned to the Air Force Satellite Control Facility at Sunnyvale AFS, California. He performed duty as the Chief of Spacecraft Operations in the Vehicle Operations office, VOF until October 1982 when he had a permanent change of assignment to Operating Division 1, Secretary of the Air Force, Special Projects. There he performed duty as Flight Mission Planning Officer until he entered the School of Engineering, Air Force Institute of Technology in May 1985. He is a member of Eta Kappa Nu and Tau Beta Pi.

Permanent address: 7301 E 33rd St
Tucson, Arizona 85710

END

4-87

DTIC